Stat 245 – Prediction Plot, Annotated

S. DeRuiter

September 16, 2022

Note: Color coding carries through entire document

Data & Plan

This example will use a dataset from fivethirtyeight.com on candy rankings.

They did an experiment (described at:

https://fivethirtyeight.com/features/the-ultimate-halloween-candy-power-ranking/)

where people had to choose between two kinds of candy.

They provide a public dataset on 85 different types of candy; for each one, a number of variables were recorded, including it's win percentage in the experimental duels.

Variable definitions are at **https://github.com/fivethirtyeight/data/tree/master/candy-power-ranking**.

dataset name

The planning phase of this modeling exercise is super abbreviated since the aim is simply to fit any old model so that the process of producing a prediction plot "by hand" can be presented. The dataset is called `candy`.

My response variable is `winpercent`, the percentage of head-to-head contests in which online survey participants said that candy was the better of the pair they were presented with. According to our rule of thumb $p < \frac{n}{15}$, we can estimate about 5 parameters, given the size of the dataset. Based on extensive experience eating candy, I suspect that `chocolate`, `peanutyalmondy`, `caramel`, `hard`, and `pluribus` will be important predictors of how much a candy is loved. I guess that `bar` and `pluribus` will contain a lot of the same information (since many candies are one or the other), so I choose just one - and it's my guess that sometime people might prefer a handful of small candies. I don't think sugariness is a big determinant of deliciousness, nor that price is the main thing that helps people choose what candy they love most.

I won't be showing prediction plots for *all* predictors, just one (though in a full analysis of course I'd be likely to do them all). I fit a model with all my predictors using `lm()`, and show a summary of the fitted model:

fitted model

```
candy_win_mod <- lm(winpercent ~ chocolate + caramel +
                      peanutyalmondy +  hard + pluribus ,
                    data = candy)
summary(candy_win_mod)
```

NAME of input to an R function (can never be changed!)

```
##
## Call:
## lm(formula = winpercent ~ chocolate + caramel + peanutyalmondy +
##     hard + pluribus, data = candy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.3913  -8.2879  -0.1437   7.6721  24.5822
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)         42.948      2.805  15.310  < 2e-16 ***
## chocolateyes        15.217      2.969   5.126 2.05e-06 ***
## caramelyes           2.298      3.493   0.658   0.5125
## peanutyalmondyyes    7.354      3.601   2.042   0.0445 *
## hardyes             -3.344      3.466  -0.965   0.3375
## pluribusyes         -0.493      2.696  -0.183   0.8554
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.28 on 79 degrees of freedom
## Multiple R-squared:  0.4477, Adjusted R-squared:  0.4127
## F-statistic: 12.81 on 5 and 79 DF,  p-value: 4.013e-09
```

The equation of the fitted model is:

$$y = 42.95 + 15.22 I_{choc} + 2.30 I_{caramel} + 7.35 I_{nuts} - 3.35 I_{hard} + 0.49 I_{pluribus} + \epsilon,$$

Where:

- I_{choc} is an indicator variable that is 0 when chocolate is "no", and 1 when it is "yes". *Note that this is exactly what the variable WAS before I changed it from numeric 0-1 coding to words! This is why, when you have just two options for a categorical variable, the 0-1 numeric representation and the string one are perfectly equivalent.
- $I_{caramel}$ is an indicator variable with value 1 if candy contains caramel, and 0 otherwise
- I_{nuts} is an indicator variable with value 1 if candy contains peanuts or almonds, and 0 otherwise
- I_{hard} is an indicator variable with value 1 if hard candy, and 0 otherwise
- $I_{pluribus}$ is an indicator variable with value 1 if many pieces, and 0 otherwise
- $\epsilon \sim N(0, 11.28)$ (The residuals ϵ follow a normal distribution with mean 0 and standard deviation 11.3.)

Prediction Plot

The code below shows how to make the plots "by hand". With this dataset, we have to be a bit careful to avoid impossible combinations - for example, `bar` AND `pluribus`.

Generate hypothetical data

name of variable being created

```
choc_pred_data <- expand.grid(chocolate = c('no', 'yes'),
                              fruity = 'no',
                              caramel = 'no',
                              peanutyalmondy = 'no',
                              nougat = 'no',
                              crispedricewafer = 'no',
                              hard = 'no',
                              bar = 'no',
                              pluribus = 'yes',
                              sugarpercent = 0.465,
                              pricepercent = 0.465
                              )
```

fake dataset #1

Make predictions from fitted model

```
choc_preds <- predict(candy_win_mod,
                      newdata = choc_pred_data,
                      se.fit = TRUE)
```

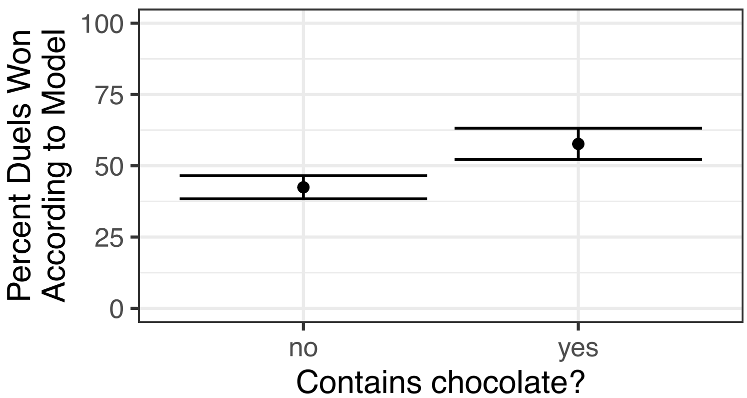
temporary storage for predictions

Put predictions, SEs, and CI in fake dataset

```
choc_pred_data <- choc_pred_data |>
  mutate(fitted = choc_preds$fit,
         CI_low = fitted - 1.96 * choc_preds$se.fit,
         CI_up = fitted + 1.96 * choc_preds$se.fit)
```

Graph the result

```
gf_point(fitted ~ chocolate, data = choc_pred_data) |>
  gf_labs(x = 'Contains chocolate?',
          y = 'Percent Duels Won\nAccording to Model') |>
  gf_lims(y = c(0,100)) |>
  gf_errorbar(CI_low + CI_up ~ chocolate)
```



There is a clear increase in winningness - from about 40 to 60 percent wins - for candy with chocolate. Even taking uncertainty in coefficient estimates into account, there is a definite difference; the CIs do not even overlap.

Now, this is silly, since `pricepercent` is not even in our model. But the code below shows a model for how to make a prediction plot *if you have a quantitative predictor*. And, we should be able to verify that the predicted "trend" is a horizontal line.

Make hypothetical dataset

fake data #2

```
price_pred_data <- expand.grid(chocolate = 'no',
                               fruity = 'no',
                               caramel = 'no',
                               peanutyalmondy = 'no',
                               nougat = 'no',
                               crispedricewafer = 'no',
                               hard = 'no',
                               bar = 'no',
                               pluribus = 'yes',
                               pricepercent = seq(from = 0, by = 0.025, to = 1),
                               sugarpercent = 0.465)
```

Make predictions

```
price_preds <- predict(candy_win_mod,
                       newdata = price_pred_data,
```

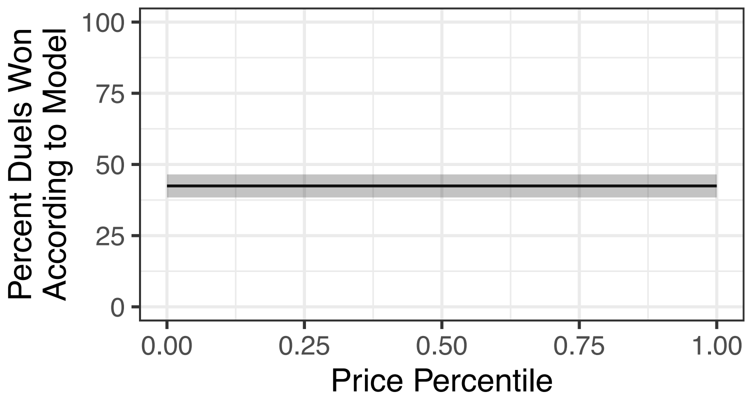
```
                         se.fit = TRUE)
```

Put predictions, SEs, and CI into fake dataset

```
price_pred_data <- price_pred_data |>
  mutate(fitted = price_preds$fit,
         CI_low = fitted - 1.96 * price_preds$se.fit,
         CI_up = fitted + 1.96 * price_preds$se.fit)
```

Graph the results

```
gf_line(fitted ~ pricepercent, data = price_pred_data) |>
  gf_labs(x = 'Price Percentile', y = 'Percent Duels Won\nAccording to Model') |>
  gf_lims(y = c(0,100)) |>
  gf_ribbon(CI_low + CI_up ~ pricepercent)
```



As expected, our model predicts absolutely no effect of price percentile on winningness (it was NOT even in the model. . .)